Estimating Cost of Software Development with Traditional and Hesitant Fuzzy Pairwise Comparison Methods: An Application

Ayfer Başar*
Istanbul Technical University, Industrial Engineering Department, Macka Campus, 34357, Istanbul, Turkey.

* Corresponding author. Tel.: +90 02124846387; email: ayferbasar@gmail.com
Manuscript submitted November 30, 2016; accepted December 30, 2016.
doi: 10.17706/jsw.12.5.355-363

Abstract: Estimating the cost of software is a fundamental and difficult process in information technology (IT) industry. Underestimation causes assigning fewer resources than the project needs and setting an unrealistic schedule. On the other hand, overestimation results in waste of resources and low customer satisfaction. It is clear that, software projects can be prioritized efficiently and resources can be used effectively by using efficient estimation methods. This paper presents a Traditional Pairwise Comparison (TPC) which involves a number of factors selected with the help of expert judgments working in a Turkish IT company. There may be uncertainties while evaluating the relative importance of two factors with TPC. Therefore, Hesitant Fuzzy Pairwise Comparison (HFPC) using Hesitant Multiplicative Geometric (HMG) operator is also presented to estimate cost of software projects having uncertainty in judgment. Both methods are applied to a software cost estimation problem for a Turkish IT company. It is seen that both methods provide efficient estimations in comparison with the real effort.

Key words: Software cost estimation, pairwise comparison methods, hesitant multiplicative geometric operator.

1. Introduction

Software development has become a fundamental factor in competitive strength of companies in recent years. Therefore, companies need to plan required time for developing applications. Estimated costs are important for both software engineers and customers. Accuracy of estimation is crucial because development projects are classified and prioritized according to the plan made by the estimation. Furthermore, resources are used more effectively, change requests can be better managed and projects can easily be controlled with accurate estimation [1]. Software development costs can be estimated by effort in man-hour/day/months, and in monetary value.

This study proposes TPC and HFPC with HMG operator to identify the importance of the cost factors and estimate the cost of the software development projects. For this aim, firstly the most important objects which cover all the entities developed for a software project and affect cost estimation are selected by the experts as the number of client (screen, user interface), application (facade, entity), batch, database (stored procedure, user-defined function), and data (table, field, sequence) objects. These objects are also addressed for the software development studies in the literature [2] - [5].

In the second step, priorities of these five objects are identified using TPC and HFPC with HMG operator.

Since experts' judgment is fundamental in terms of estimation, a subjective weighting technique is preferred to find the importance of the cost factors instead of using an objective weighting technique (i.e., Regression Analysis). Since decision makers focus on only two alternatives while making judgment, Pairwise Comparison is found a more efficient method among various subjective weighting techniques (i.e., Rating, Point Allocation, Ratio, Ranking, and Trade-off) [6]. Furthermore, Pairwise Comparison generally gives more accurate results in comparison with the other weighting methods [7]. However, there may be uncertainties while evaluating the relative importance of two factors with expert opinion. For instance, the decision maker may evaluate the importance of one factor over another as "about 3", "between 2 and 3", "at least 4" or "at most 5" times etc. Therefore, HFPC is used to determine the importance of factors by coping with this uncertainty in judgment. HMG operator is used to aggregate experts' judgment in HFPC method. Both TPC and HFPC methods are applied in estimating cost of 1,180 real software development projects of a Turkish IT company which are completed and the deviation between the real and estimated effort for all the projects are calculated. It is seen that both methods provide efficient estimations. The results are also approved by the experts and managers who contributed in identifying the cost factors and specifying their importance.

The rest of the paper is organized as follows: Section 2 is dedicated to the literature review about cost estimation methods. In Section 3, the concepts of Pairwise Comparison including fuzzy and hesitant fuzzy extensions are discussed. The application of the proposed methodology in a Turkish IT company is presented in Section 4. Finally, conclusions and further research are provided in Section 5.

2. Cost Estimation Techniques

Although there are lots of cost estimation methods for software development in the literature, there are two categories basically: algorithmic and non-algorithmic. Expert judgment is the most common non-algorithmic method which involves asking experts who are experienced on the projects. However, the level of accuracy in this technique is quite low because of biased or inconsistent evaluation in most cases. [8] presents overconfidence in judgmental forecasting. In order to overcome the inconsistencies in expert judgment, various techniques such as Delphi [9], [10] and Pairwise Comparisons [11] are used in the literature. As another non-algorithmic method used to estimate software development costs, Parkinson's principle determines the cost depending on resources and delivery time [12]. Bottom-up and top-down approaches are other examples of non-algorithmic methods. In bottom-up technique, after estimating the cost of all the components of the software, final estimation of the entire project is found by aggregation. This technique requires a detailed analysis to be completed before estimation. Conversely, top-down approach is used to split the total estimated cost of the software to its components. Thus, it is generally preferred at the early stage in the software development life cycle and does not require detailed analysis. The final non-algorithmic method is estimation by analogy. This technique uses actual costs of a completed project which is very similar to the current project. Analogy can be found an efficient method due to using actual costs rather than estimation. However, it is difficult to find a similar project in terms of scope, risk, resource usage etc. [13].

Algorithmic methods used for software cost estimation are different than non-algorithmic ones since they depend on mathematical models and find objective results. The cost of the software is estimated by using a cost function involving different factors in algorithmic methods. A simple cost function can be shown as $f(x_1, x_2, \ldots, x_n)$ where $x_1, x_2, \ldots, x_n$ represent the cost factors. Thus, decision of cost factors and the function to be used becomes vital to estimate the cost accurately. The most common cost factors are based on product (reusability, reliability, complexity); computer (execution time); personnel (experience, turnover); project (scope, risks, development environment, software tool). Algorithmic estimation techniques can be linear, multiplicative or power function. a x S^b formula is used in power function models where S shows the size of the code, a and b are the other cost factors. COCOMO (Constructive Cost Model) proposed by [14] is the most well-known power function models. COCOMO uses S as thousands of lines of code (*KLOC*) and effort as person-month. It is clear that estimating the line of codes before completing the project is a difficult task. In most instances, expert judgment along with PERT technique is used to estimate the code size. Thus, experts are expected to estimate the code size with three ways: S_l, the lowest; S_h the highest; and S_m, the most likely size. Then, the code size S is estimated by $S = (S_l + 4\ S_m + S_h) / 6$. COCOMO models are categorized in three groups: i) Basic COCOMO, ii) Intermediate COCOMO, and iii) COCOMO II. Basic COCOMO is the earliest model developed by [14] and it estimates the effort by the formula a x $(KLOC)^b$ where a and b refers the complexity of the code. a and b take values according to the complexity of the software in the Simple COCOMO: 2.4 and 1.05 if the size and complexity is simple; 3.0 and 1.15 for more complex programming activities; 3.6 and 1.20 if the software is complex, respectively. Since Basic COCOMO is based on the assumption that cost depends on only code size and does not take into consideration other factors affecting the cost, extensions of Basic COCOMO are introduced in the literature. Intermediate COCOMO obtains more accurate estimation by using all related parameters. It uses a x $(KLOC)^b$ x *EAF* where *EAF* addresses effort adjustment factor. In Intermediate COCOMO, the values of parameter b are the same as in the Simple COCOMO, however the values of parameter a are different: 3.2 for the simple, 3.0 for more complex, and 2.8 for the complex programming activities. In this approach, the product of all cost factors is calculated to find *EAF*. Basic and Intermediate COCOMO models have an important drawback that they consider a software project as a single product having similar properties although large systems may involve lots of sub-systems with different characteristics. Therefore, COCOMO II was introduced in the literature as the latest version of COCOMO methods [15]. In COCOMO II, parameter b changes depending on 5 new cost factors: process maturity, precedentedness, development flexibility, risk resolution, and team cohesion.

In recent years, agile software development approaches have been widely used due to the increase in the uncertainties about customer requirements where companies focus on small development, incremental and iterative releases different than traditional techniques. Artificial neural networks are proposed for these types of software development projects [16, 17]. Moreover, meta heuristic techniques are improved to find the best solution for cost estimation problem in software development especially for the large problems. For instance, Harmony Search [18], Tabu Search [19], Genetic [19, 20, 21], Artificial Bee Algorithms [20] are proposed to solve software cost estimation mathematical models favorably in terms of both computation time and objective function value.

3. Traditional and Hesitant Fuzzy Pairwise Comparison Methods

Software development cost estimation is a type of Multi-Criteria Decision Making (MCDM) problem since there are lots of cost factors affecting the estimation. MCDM deals with decision making with multiple criteria or objectives. In most cases, the objectives are conflicting, and it is very difficult to choose a criterion over another. Therefore, result of MCDM problems mostly depends on decision makers' options [22]. Pairwise Comparison is found to be an efficient method to find the importance of criteria in MCDM problems in the literature [23]. On the other hand, it is incompetent to cope with the imprecision and subjectivity of decision makers. In the literature, fuzzy Pairwise Comparison is presented to overcome this drawback [24]. This method uses a range of values instead of a single crisp value, thus the decision makers can select the value reflecting their judgment and specify their attitude like optimistic, pessimistic or moderate. Moreover, hesitant fuzzy sets which are used in the Pairwise Comparisons are newly introduced in the literature and there are only a few studies using these sets [25].

The Pairwise Comparison is originally proposed by [23] and used in Analytical Hierarchy Process with a hierarchical structure of criteria in the related MCDM problem. Since it is found as an effective method to find the importance of criteria by a series of expert judgments, it has been implemented for many studies in different problems such as project planning [26]; location selection [27]; software cost estimation [28] etc. The method is applied by following steps:

Step 1: The decision hierarchy, including a goal and related (sub)criteria is prepared to make judgments in pairs.

Step 2: A Pairwise Comparison matrix (A) is built to specify the importance of different criteria and sub criteria. Each entry (a_{ij}) of the matrix A refers the importance of the factor i in comparison with the factor j. The relative importance values of the criteria are found by using the scale of 1–9 corresponding to the linguistic comparisons: 1 for equal importance, 3 for moderate importance of one over another, 5 for strong or essential importance, 7 for very strong or demonstrated importance, 9 for extreme importance, and 2-4-6-8 for intermediate values. [29].

Step 3: Relative importance of the criteria $w = (w_1, w_2., w_n)$ is found by solving $Aw = nw$, which is the eigenvector of A.

Ref. [30] proposed fuzzy set theory to deal with imprecision in MDCM problems. In classical set theory an element must either be included or not to a set; however in in fuzzy set theory, an element can belong to a set to a degree k ($0 \leq k \leq 1$) and the elements are the members of the interval known as membership function. [24] introduced the method to calculate the fuzzy importance weights by allowing the decision maker to select one criterion over the other when there is uncertainty, and triangular fuzzy number is used as $a_{ij} = (l_{ij}, m_{ij}, u_{ij})$ where l refers the lower limit, m addresses the most probable and u shows the upper limit value. It is obvious that specifying the membership degree of an element is a difficult process since there may be possible values which make decision makers hesitate when making judgments in paired comparison. For this reason, hesitant fuzzy sets were introduced to solve this problem [31]. The fundamental properties of hesitant fuzzy sets are given as follows:

Definition 1: If X is a fixed set, a hesitant fuzzy set on X returns a subset of [0, 1] by:

$$E = \{ < x, h_E(x) > | \, x \in X \}; \tag{1}$$

where $h_E\,(x)$ shows the possible membership degrees of the element $x \in X$ to the set E and takes values between 0 and 1.

Definition 2: The upper and lower limit is found by:

$$h^-(x) = \min h\,(x) \tag{2}$$
$$h^+(x) = \max h\,(x) \tag{3}$$

Definition 3: Basic operations for hesitant fuzzy sets h, h_1 and h_2 are found as follows:

$$h^\lambda = \bigcup_{\gamma \in h}\{\gamma^\lambda\} \tag{4}$$
$$\lambda h = \bigcup_{\gamma \in h}\{1-(1-\gamma)^\lambda\} \tag{5}$$
$$h_1 \bigcup h_2 = \bigcup_{\gamma 1 \in h1,\, \gamma 2 \in h2} \max\{\gamma^1, \gamma^2\} \tag{6}$$
$$h_1 \bigcap h_2 = \bigcup_{\gamma 1 \in h1,\, \gamma 2 \in h2} \min\{\gamma^1, \gamma^2\} \tag{7}$$
$$h_1 \oplus h_2 = \bigcup_{\gamma 1 \in h1,\, \gamma 2 \in h2} \{\gamma^1 + \gamma^2 - \gamma^1\gamma^2\} \tag{8}$$
$$h_1 \otimes h_2 = \bigcup_{\gamma 1 \in h1,\, \gamma 2 \in h2} \{\gamma^1\gamma^2\} \tag{9}$$

Definition 4: An HMG operator is found by:

$$HMG(a_1, a_2, \ldots, a_n) = \otimes_{i=1}^{n} a_i^{\frac{1}{n}} = \cup_{\xi_1^{\sigma(s)} \in a_1, \ldots, \xi_n^{\sigma(s)} \in a_n} \left\{ \prod_{i=1}^{n} \left(\xi_i^{\sigma(s)}\right)^{\frac{1}{n}} \right\} \tag{10}$$

4. Application of Traditional and Hesitant Fuzzy Pairwise Comparison Method in a Turkish Company

In order to determine the criteria effective for the software development cost estimation, both the factors used in the literature [2]–[5] and expert opinion is considered. Ultimately, the most fundamental factors on a software development project are selected as the number of client, application, batch, database, and data objects. An unstructured questionnaire is conducted by the help of face to face meetings to find the fuzzy weights of factors. Interviews are made with six experts working in a Turkish IT company as managers and five different cost factors (number of client: f_1, application: f_2, batch: f_3, database: f_4, and data: f_5 objects) are evaluated by the experts. Table 1 shows Pairwise Comparison matrices of six experts' judgments.

Table 1. TPC of Cost Factors

Expert 1	f_1	f_2	f_3	f_4	f_5
f_1	1	3	5	9	9
f_2		1	1	5	7
f_3			1	1	5
f_4				1	5
f_5					1

Expert 2	f_1	f_2	f_3	f_4	f_5
f_1	1	5	5	1	3
f_2		1	5	1	7
f_3			1	1	3
f_4				1	9
f_5					1

Expert 3	f_1	f_2	f_3	f_4	f_5
f_1	1	3	3	1	9
f_2		1	7	7	5
f_3			1	5	3
f_4				1	9
f_5					1

Expert 4	f_1	f_2	f_3	f_4	f_5
f_1	1	5	9	0,33	5
f_2		1	5	1	5
f_3			1	1	1
f_4				1	3
f_5					1

Expert 5	f_1	f_2	f_3	f_4	f_5
f_1	1	2	7	5	9
f_2		1	3	9	3
f_3			1	1	1
f_4				1	7
f_5					1

Expert 6	f_1	f_2	f_3	f_4	f_5
f_1	1	7	9	3	7
f_2		1	1	0,2	2
f_3			1	0,2	1
f_4				1	5
f_5					1

In the second step, experts' judgments in TPC are collaborated by the help of geometric mean and a pairwise comparison matrix (A) is built. The results are presented in Table 2. As an example, relative importance of the first factor over second factor is calculated by $(3 \times 5 \times 3 \times 5 \times 2 \times 7)^{(1/6)} = 3.83$.

Table 2. Collaboration of TPC

	f_1	f_2	f_3	f_4	f_5
f_1	1.00	3.83	5.91	1.88	6.52
f_2	0.26	1.00	2.84	1.99	4.41
f_3	0.17	0.35	1.00	1.00	1.89
f_4	0.53	0.50	1.00	1.00	5.91
f_5	0.15	0.22	0.53	0.17	1.00

In the final step of TPC, relative importance weights of the criteria $w = (w_1, w_2 \ldots, w_n)$ is found by solving $Aw = nw$ and the results are given in Table 3.

Table 3. Eigenvector and Weights Found by TPC

	Eigenvector					Weight
	f_1	f_2	f_3	f_4	f_5	
f_1	0.474	0.649	0.524	0.311	0.330	0.458
f_2	0.123	0.169	0.252	0.330	0.224	0.220
f_3	0.080	0.059	0.089	0.165	0.096	0.098
f_4	0.251	0.085	0.089	0.165	0.300	0.178
f_5	0.072	0.038	0.047	0.028	0.051	0.046

As seen in Table 3, number of client objects (f_1) is the most important factor with its weight 45.8%. Number of application objects (f_2) has also high importance weight, 22%. Number of batch (f_3) and database objects (f_4) has significance by 9.8% and 17.8%, respectively. Finally, number of data objects (f_5) is the least important factor on the cost estimation.

In order to find the importance of cost factors with HFPC method, a collaborative Pairwise Comparison matrix is constructed by using individual opinions of six experts. Table 4 represents aggregation of 6 experts' judgments.

Table 4. Aggregation of Experts' Judgments

6 Experts	f_1	f_2	f_3	f_4	f_5
f_1	1	{2, 3, 5, 7}	{3, 5, 7, 9}	{0.33, 1, 3, 5, 9}	{3, 5, 7, 9}
f_2	{0.14, 0.2, 0.33, 0.5}	1	{1, 3, 5, 7}	{0.2, 1, 5, 7, 9}	{2, 3, 5, 7}
f_3	{0.11, 0.14, 0.2, 0.33}	{0.14, 0.2, 0.33, 1}	1	{0.2, 1, 5}	{1, 3, 5}
f_4	{0.11, 0.2, 0.33, 1, 3}	{0.11, 0.14, 0.2, 1, 5}	{0.2, 1, 5}	1	{3, 5, 7, 9}
f_5	{0.11, 0.14, 0.2, 0.33}	{0.14, 0.2, 0.33, 0.5}	{0.2, 0.33, 1}	{0.11, 0.14, 0.2, 0.33}	1

All the combinations of the cost factors are determined in order to apply HFPC for the aggregated judgments given in Table 4. It is clear that there are 1 x 4 x 4 x 5 x 4 = 320 combinations for f_1, 4 x 1 x 4 x 5 x 4 = 320 combinations for f_2, 4 x 4 x 1 x 3 x 3 = 144 combinations for f_3, 5 x 5 x 3 x 1 x 4 = 300 combinations for f_4, and 4 x 4 x 3 x 4 x 1 = 192 combinations for f_5. For instance, {0.11, 0.14, 1, 0.2, 1} is one of the 144 combinations for f_3. According to the HMG operator, aggregated weight for this combination is computed by $(0.11^{0.2} \times 0.14^{0.2} \times 1^{0.2} \times 0.2^{0.2} \times 1^{0.2}) = 0.315$ since there are 5 different cost factors, thus $n = 5$ and $1/n = 0.2$. By applying similar calculations to all the cost factors, hesitant fuzzy weights of f_1, f_2, f_3, f_4, and f_5 are found as in Table 5. Finally, fuzzy weights of cost factors are normalized and the criteria are ranked according to their normalized weights in Table 5.

Table 5. Hesitant Fuzzy Weights of Cost Factors

Factor	Hesitant Fuzzy Weights	Normalized Weights
f_1	3.134	0.455
f_2	1.578	0.229
f_3	0.711	0.103
f_4	1.142	0.166
f_5	0.327	0.047

It is clear that there is only slight difference between the results found by TPC in Table 3 and HFPC in Table 5 with 45.5%, 22.9%, 10.3%, 16.6% and 4.7% hesitant fuzzy normalized weights, respectively for f_1, f_2, f_3, f_4, and f_5.

Finally, in order to evaluate the efficiency of TPC and HFPC methods, both techniques are applied to the software development cost estimation problem of a Turkish IT company (the name of the company cannot be given because of the confidentiality reasons,) based on the real data. For this aim, firstly, values of f_1, f_2, f_3, f_4, and f_5 for 1,180 completed projects are specified. Subsequently, costs are estimated by using weights in Tables 3 and 5 and real values of f_1, f_2, f_3, f_4, and f_5 for each project. Later, effectiveness of TPC and HFPC is analyzed by finding the deviation between the actual and estimated effort. Table 6 shows the results of TPC and HFPC in comparison with real development costs in man-day.

Table 6. Benchmark of Real Costs and Estimations Found by TPC and HFPC for 1,180 Projects

Interval of Real Costs (x) (man – day)	Number of Projects	Average Real Cost (man–day)	Average Estimated Cost Found by TPC (man–day)	Average Estimated Cost Found by HFPC (man–day)	Deviation of TPC (%)	Deviation of HFPC (%)
$0 < x < 10$	164	6,74	5,98	6,13	11,28%	9,05%
$10 \leq x < 50$	593	35,26	28,49	30,28	19,20%	14,12%
$50 \leq x < 100$	358	64,08	74,85	73,04	-16,81%	-13,98%
$x \geq 100$	65	192,32	226,09	219,64	-17,56%	-14,21%

The deviation in Table 6 is calculated by the formula (1 - Estimated / Real Cost). Thus, the positive deviation means that the average estimated cost for the projects is lower than the average real effort. As seen in Table 6, the estimated cost obtained by TPC is 11.28% and 19.20% lower than for the projects costs of which are smaller than 10 and 10-50 man-days on the average, respectively. On the other hand, estimated costs are higher than real effort by 16.81% and 17.56% for the projects cost of which is 50-100 and higher than 100 man-days on the average, respectively. Furthermore, the estimated cost obtained by HFPC is 9.05% and 14.12% lower than for the projects costs of which are smaller than 10 and 10-50 man-days on the average, respectively, while estimated costs are higher than real effort by 13.98% and 14.21% for the projects cost of which is 50-100 and higher than 100 man-days on the average. This shows that, both methods provide efficient estimations with low deviation from the real development cost. However, HFPC estimates software development cost more effectively than TPC depending on the number of f_1, f_2, f_3, f_4, and f_5 in the projects although there is a slight difference between the normalized weights of the cost factors. Experimental study shows that estimated costs found by TPC and HFPC methods are lower than real costs for the projects with less than 50 man – days effort while they are higher than real effort for the larger projects. Estimations of the small projects are found to be more acceptable by the experts working as executives in the company who are responsible for the projects and by the managers who contributed in finding the weights of the the factors. Thus, it is seen that TPC and HFPC are efficient for especially small projects, and they also provide meaningful estimations for the large projects. Consequently, it may be more beneficial to estimate costs by using TPC and HFPC for the small software development projects or large ones which are divided into the phases.

5. Conclusion

This study presents methods to solve software development cost estimation problem. Analytical methods and expert opinion are commonly used ways for this purpose, but both approaches have some disadvantages. Analytical methods are incompetent while considering expert judgment, and the accuracy depends on only cost function and factors used. Moreover, estimation found by expert judgment without using analytical methods may not be accurate in case experts are not unbiased and experienced enough.

In this study, we derive benefit from the strengths and discard the insufficiencies of analytical methods and expert judgment by integrating them. Studies related to fuzzy approaches to estimate software development costs in an uncertain environment are scarce in the literature. In this paper, firstly we determine the most effective cost factors via expert opinion and literature survey, and then we obtain importance weights of cost factors by TPC and HFPC using expert opinion. We experimentally show that TPC and HFPC methods give efficient solutions for 1,180 projects completed and especially more reliable estimations for projects requiring less than 50 man – day effort. To sum up, estimating costs by TPC and HFPC can be substantial for small or multi-phase projects.

In the future, a mathematical model using weights obtained by TPC and HFPC and different constraints (i.e., maximum number of developers responsible for each project, meeting deadline desired by the customer etc.) can be developed to find the optimum solution of the problem.

References

[1] Aguilar-Ruiz, J. S., Ramos, I., Riquelme, J. C. & Toro, M. (2001). An evolutionary approach to estimating software development projects. *Information and Software Technology*, *43(14)*, 875-882.

[2] Jacobson, I. (1993). Object-oriented software engineering: A use case driven approach. Wokingham, Berkshire, UK: Addison-Wesley.

[3] Ribu, K. (2001). Estimating object-oriented software projects with use cases. M.S. thesis, University of

Oslo, Oslo, Norway.

[4] Nesi, P. & Querci, T. (1998). Effort estimation and prediction of object-oriented systems. *Journal of* Systems *and Software*, *42*, 89-102.

[5] Antoniol, G., Fiutem, R., & Lokan, C. (2003). Object-oriented function points: An empirical validation. *Empirical Software Engineering*, *8(3)*, 225-254.

[6] Malczewski, J. (1999). GIS and multi criteria decision making. New York, NY, USA: John Wiley and Sons.

[7] Badri, M. A. (2001). A combined AHP – GP model for quality control systems. *International Journal of Production Economics*, *72*, 27–40.

[8] Lin, S. W. & Bier, V. M. (2008). A study of expert overconfidence, *Reliability Engineering and System Safety*, *93*, 711-721.

[9] Stellman, A., & Greene, J. (2005). Applied software project management. O'Reilly Media.

[10] Wiegers, K. (2000). Stop promising miracles. *Software Development*, *8*, 49 –53.

[11] Miranda, E. (2001). Improving subjective estimates using paired comparisons. *Software.* 87-91.

[12] Parkinson, G. N. (1957). *Parkinson's Law and other studies in administration*. Boston, USA: Houghton-Miffin.

[13] Shepperd, M., & Schofield, C. (1997). Estimating software project effort using analogy. *IEEE Transactions on Software Engineering*, *23*, 736-743.

[14] Boehm, B. W. (1981). Software engineering economics. USA: Prentice-Hall.

[15] Boehm, B. W. (2000). Software cost estimation with COCOMO II. Englewood Cliffs, NJ, USA: Prentice Hall.

[16] Bilgaiyan, S., Mishra, S., & Das, M. (2016). A review of software cost estimation in agile software development using soft computing techniques. *Proceedings of the 2016 2nd International Conference on Computational Intelligence and Networks (CINE)*.

[17] Sarno, R., Sidabutar, J., & Sarwosri. (2015). Comparison of different neural network architectures for software cost estimation. *Proceedings of the 2015 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*.

[18] Jafari, S. M. S., & Ziaaddini, F. (2016). Optimization of software cost estimation using harmony search algorithm. *Proceedings of the 2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*.

[19] Gharehchopogh, F. S., Rezaii, R., & Arasteh, B. (2015). A new approach by using tabu search and genetic algorithms in software cost estimation. *Proceedings of the 2015 9th International Conference on Application of Information and Communication Technologies (AICT)*.

[20] Gharehchopogh, F. S., Maleki, I., & Talebi, A. (2015). Using hybrid model of artificial bee colony and genetic algorithms in software cost estimation. *Proceedings of the 2015 9th International Conference on Application of Information and Communication Technologies (AICT)*, Rostov on Don.

[21] Benala, T. R., Dehuri, S. C., Satapathy, S. C., & Raghavi, C. S. (2011). Genetic algorithm for optimizing neural network based software cost estimation. *Proceedings of the International Conference on Swarm, Evolutionary, and Memetic Computing*.

[22] Hwang, C. L. & Yoon, K. (1981). Multiple attributes decision making methods and applications. Berlin, Germany: Springer.

[23] Saaty, T. L. (1980). The analytic hierarchy process. New York, NY, USA: McGraw – Hill Inc.

[24] Buckley, J. J. (1985). Fuzzy hierarchical analysis. *Fuzzy Sets and Systems*.

[25] Zhu, B., & Xu, Z. (2014). Analytic hierarchy process-hesitant group decision making. *European Journal of* Operational *Research*, *239*, 794– 801.

[26] Oztaysi, B. (2014). A decision model for information technology selection using AHP integrated

TOPSIS-Grey: The case of content management systems. *Knowledge-Based Systems*, *70*, 44-54.

[27] Basar, A. Kabak, O., Topcu, Y. I. & Bozkaya, B. (2014). Location analysis in banking: A new methodology and application for a Turkish bank. *Applications of Location Analysis.*

[28] Sivakumar, R., Kannan, D., & Murugesan, P. (2015). Discussion of green vendor evaluation and selection using AHP and Taguchi loss functions in production outsourcing in mining industry. *Resources Policy*.

[29] Saaty, T. L. (1990). How to make a decision: The analytic hierarchy process. *European Journal of Operational Research*, *48*, 9–26.

[30] Zadeh, L. (1965). Fuzzy sets. *Information and Control*, *8*, 338-353.

[31] Torra, V. (2010). Hesitant fuzzy sets. *International Journal of Intelligent Systems*, *2*5, 529–539.

Ayfer Basar was born in Istanbul, Turkey. She received her B.Sc. and Ph.D. degrees from Istanbul Technical University (ITU) in 2006 and 2014 and M.Sc degree from Sabanci University, Istanbul in industrial engineering in 2008. She is a part-time instructor in Ozyegin University, Istanbul. Her research interests are IT process optimization, cost estimation, location planning, and meta heuristic approaches.

She is a service manager in one of the biggest national bank's IT company in Turkey for 9 years. She has published lots of articles in indexed journals such as Journal of the Operational Research Society, Optimization Letters, Journal of Information Technology and Decision Making.